SC-T-213-VEFF - Web Programming I

Project Assignment 4 – Endpoint Testing

Reykjavik University | Deadline: **7th April 2021, 23:59**

The topic of this assignment is: **Writing endpoint tests for a RESTful Backend**.

1 Overview

In this assignment, you will write endpoint tests that test a modified solution to Assignment 3 (boards & tasks backend). This will include the user of authentication/authorization.

This assignment is individual, no group work is permitted.

To get started, you are encouraged to try out all endpoints using Postman.

2 Setup

In the supplementary material, you find a modified solution to Assignment 3. It has fewer endpoints, includes authentication, has a reset method (to facilitate testing), and lacks sorting of tasks. The basic 7 endpoints are as follows:

1. **GET /api/v1/boards**
2. **GET /api/v1/boards/:boardId**
3. **POST /api/v1/boards**
4. **PUT /api/v1/boards/:boardId**
5. **GET /api/v1/boards/:boardId/tasks**
6. **GET /api/v1/boards/:boardId/tasks/:taskId**
7. **POST /api/v1/boards/:boardId/tasks**

These endpoints have the same functionality/fulfil the same requirements as in assignment 3, with the exception that the sorting functionality is lacking for tasks.

Additionally, there are two more endpoints:

8. **POST /api/v1/auth**
9. **DELETE /api/v1/boards/:boardId**

The first endpoint expects an empty body and a correct HTTP Basic Authentication with username *admin* and password *secret*. When the request is correct, a JSON response is sent that contains an object with a single attribute called *token*.

The second endpoint has the same functionality as in assignment 3 (to delete an individual board). However, it expects a correct token (obtained from the /api/v1/auth endpoint) sent as a Bearer Authorization. This means the request needs to include the *Authorization* header with value *Bearer <token>*. Another explanation of the Bearer authorization can be found, e.g., here: https://swagger.io/docs/specification/authentication/bearer-authentication/.

In the *test* sub-folder, you will find a file called *index.test.js*, already set up so that you can start writing your tests. The current setup makes sure that the reset method in index.js is called before each test. This means you always have two boards and one task, and you know the exact value of all their properties (see the reset function for details). You should use this knowledge when writing assertions in your tests (e.g., you know the names and descriptions of both boards, so you know what a "get all boards" request should return). Currently, the file includes a single test.

A number of npm scripts have been included, so that you can easily run the server, the tests, and the debugger[1]:

- *npm start* starts the server
- *npm test* runs all tests in *test/index.test.js* (and all other files in that folder that end on *.test.js*)
- *npm debug* starts the server in debug mode on port 9229 (compatible with VSCode *Launch via NPM*).

3 Task

Your task in this assignment is to write a number of endpoint tests. 8 points are awarded for testing of the 7 basic endpoints, 2 points are awarded for successfully testing the delete board endpoint.

3.1 Basic Endpoint Tests

(A total of 8 tests is required here)
For each basic endpoint, write a test that captures the success case (the request succeeds, resulting a 2xx response code). For endpoints that return arrays, assert the following:

- The status code shall be as expected (e.g., 200 for endpoint 1)
- The response body is in json format
- The return type is an array
- The array contains the right amount of elements

For endpoints that return individual objects, assert the following:

- The status code shall be as expected (e.g., 200 for endpoint 1)
- The response body is in json format
- The response body is as expected
  - The right attributes are in the body

[1]The npm test script might not work under Windows. Try replacing the package.json file with the provided package.json.windows file to get it to work!

- No additional attributes are in the body
- All attributes have the expected values

For POST requests, you may exclude checking the values of the id property for boards and tasks, and the dateCreated property for tasks.

In addition to the 7 success cases, write one test that ensures that PUT /api/v1/boards/:boardId does not succeed when a property is missing. You can choose which property to exclude in the request. Assert the following:

- The status code shall be 400
- The response body is in json format
- The error message is as expected.

3.2 Delete Endpoint Test

(A single test is required here)
Write one test for the **DELETE /api/v1/boards/:boardId** endpoint. The test shall test the success case. That is, a DELETE request with the right authentication information is successful. To do so, you first need to make a successful POST request to the /api/v1/auth endpoint to obtain a token. To assert that the delete request has been successful, it is sufficient to check for a 200 response code in that request.

Note: For obtaining a Base64 encoding in Node.js, you can use the following command:

```
Buffer.from("stringToBeHashed", "binary").toString("base64");
```

4 Requirements

The following requirements/best practices shall be followed:

1. The backend code shall remain unchanged.
2. No extra files shall be added. All test code shall be added in test/index.test.js
3. The tests shall be written using the mocha, chai and chai-http modules.
4. There are no restrictions on the ECMAScript (JavaScript) version.

Submission

The lab is submitted via Canvas. Submit a zip file containing your test file. Further files are not necessary since we will use the provided backend code. In particular, do **NOT** include the *node_modules* folder and the *package-lock.json*.